NPS-CS-11-008

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

BULK EXTRACTOR WINDOWS PREFETCH DECODER

by

Luis E. Garcia II

August 26, 2011

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Daniel T. Oliver
President

Leonard A. Ferrari
Executive Vice President and
Provost

This report was prepared for and funded by the Defense Intelligence Agency, Washington, DC.

This report was prepared by:

Luis E. Garcia II
Department of Computer Science

Reviewed by:

Peter Denning
Chairman
Department of Computer Science

Released by:

Karl A. van Bibber, Ph.D.
Vice President and Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
26–8–2011	Technical Report	2011-06-30—2011-08-31

4. TITLE AND SUBTITLE

Bulk Extractor Windows Prefetch Decoder

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Luis E. Garcia II

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

NPS-CS-11-008

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

DIA

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this report are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

14. ABSTRACT

scan_winprefetch is a C++ and thread-safe Windows prefetch scanner for the *bulk_extractor* framework that decodes prefetch files. The decoder analyzes disk images for Windows prefetch files. At the completion of analyzing each prefetch file found on the disk image, a text file is created containing a XML output detailing all found prefetch files.

15. SUBJECT TERMS

Carving, Microsoft Windows Prefetch Files, Bulk Extractor

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
U	U	U	UU	13	

THIS PAGE INTENTIONALLY LEFT BLANK

Abstract

scan_winprefetch is a C++ and thread-safe Windows prefetch scanner for the *bulk_extractor* framework that decodes prefetch files. The decoder analyzes disk images for Windows prefetch files. At the completion of analyzing each prefetch file found on the disk image, a text file is created containing a XML output detailing all found prefetch files.

1 Introduction

Prefetch files are files created by the Microsoft Windows operating system that contains, for each application run, the name of the application, the DLLs and directories used, application last run time, the total number of times that the application has been run, and the time that the prefetch file itself was created [1].

Despite the usefulness of this information for forensic examination, and despite support for decoding the prefetch file format in tools such as Guidance Software's Encase, the analysis of Windows Prefetch files is still a relatively rare occurrence for many computer forensics examinations.

Prefetch analysis may be relatively uncommon because the Windows operating system prunes the files in the `%SYSTEMROOT%\Prefetch` directory when the directory becomes too large. The directory entries are then soon re-allocated to other prefetch files. Thus, many of the most useful Windows prefetch files on a typical hard drive under examination will be found not in the `%SYSTEMROOT%\Prefetch` directory, but in unallocated space on the hard drive, where they can only be found through file carving.

scan_winprefetch is a C++ and thread-safe Windows prefetch scanner for the *bulk_extractor* framework that locates prefetch files in bulk data and decodes their contents. The scanner creates a block of XML containing the decoded contents for each prefetch file found, and stores this XML in the *bulk_extractor* feature file.

Windows prefetch files have an abundant amount of information that can help in answering questions of indent and suspicious behaviors of an application. But most prefetch analyzers only operate on files. By creating a prefetch decoder for *bulk_extractor*, the scanner combines file carving with prefetch decoding, potentially giving the examiner access to significantly more information.

Currently most analysis that is performed on prefetch files is a *first-order* analysis where the files are examined for plain value of the information that they contain. By outputting the contents of the decoded prefetch file in XML, the prefetch scanner also makes it easier to develop automated processing routines that perform *second-order* forensic analysis—for example, correlation between the contents of different prefetch files on the same disk, or on different disks.

By combining the prefetch scanner with *bulk_extractor*, this project enables systematic examination of prefetch files to become a standard part of all forensic examinations, with no additional time expended on the part of the examiner. The prefetch scanner imposes minimal overhead on *bulk_extractor*'s processing time.

2 Background and Related Work

2.1 *bulk_extractor*

Prefetch related research has not been wide spread since its introduction in Windows XP. As this feature continues to be unnoticed, the forensic benefit is not being used. There have been multiple discussions about prefetch files on blogs and news articles.

Windows XP introduced the methodology of pre-fetching pages and files for startup applications [5]. Prefetching improves the start-up (booting) process of an application by prefetching the necessary files and loading them into memory for use. Applications in the Windows Operating system since XP will create a prefetch file located in %SYSTEMROOT%\Prefetch. Naming convention of the file is (progname)-(hash).pf. The file will only be created during the initial run of an application. The number of files created can be up to 128 files [4]. In the file name, the hash value represents the location of where the application was run. The hash algorithm was researched by Kharti [11]. The prefetch file contains the name of the application, the DLLs and directories used, last run time, the number of times an application was run and creation time [1]. Morrison describes the Windows "Prefetcher," the program that creates prefetch files [6].

Wade discussed the forensic value of prefetch files. In addition, he discussed the properties of the prefetch, what artifacts can be used and several third party tools that he used to analyze it [1].

Kornblum presented information about the Windows Vista super prefetch file at the 2008 DoD Cyber Crime Conference [4].

Garfinkel introduced a multi-threaded bulk data analysis tool, *bulk_extractor*, that can extract forensic evidences more quickly surpassing Guidance Software's EnCase extraction speed [8]. He proposed that existing tools do not completely analyze disk images and may overlook valuable digital artifacts.

3 *scan_winprefetch*

The scanner is part of the *bulk_extractor* framework as a scanner. The scanner receives data in a *sbuf* and reports its findings using the *feature_recorder* system.

The first step of the scanner is to look for the signature of bytes that indicates the start of a prefetch file. The prefetch file signature is an eight byte value. In Windows XP, Vista and 7, the signature is different by a single byte. On Windows XP, the signature has the first byte to be 0x11 (17 in decimal) (figure 1). On Windows Vista and Windows 7, the 0x11 is replaced by 0x17 (23 in decimal) (figure 2).

Once a header is found, the scanner creates a C++ object that contains a reference to the prefetch file's start in the bulk data. The object contains instance methods that decode and return on-the-fly other information within the prefetch file, including the last run time, position of DLLs and directories, end of file offset, etc. The scanner is designed so that attempts to reference memory outside the buffer result in a return value of "0." This would happen if the prefetch file is truncated or corrupt.

```
00000000  11 00 00 00 53 43 43 41  0f 00 00 00 3e a3 01 00  |....SCCA....>...|
```

Figure 1: Windows XP prefetch file signature

The scanner automatically decodes the filenames from the UTF-16 encoding stored in prefetch files and

```
00000000  17 00 00 00 53 43 43 41  11 00 00 00 3a 2a 05 00  |....SCCA....:*..|
```

Figure 2: Windows 7 prefetch file signature

re-encodes the filenames as UTF-8 which is stored in the feature file. Characters that would result in an invalid Unicode coding are ignored, as they are usually the result of corruption in the disk image. Times are presented in ISO 8601 format.

In typical use, multiple prefetch files are found, and they are all recorded in the single *winprefetch.txt* feature file. The features are encoded using a set of Digital Forensics XML tags that represent feature file artifacts [2].

4 Results

A prefetch file may have many DLLs and directories associated to it. The number of DLLs and directories used depends on the applications startup process. Figure 3 is a sample of the XML produced by *scan_winprefetch*. The XML has been pretty-printed for ease of viewing.

Each individual prefetch file is enclosed in `<prefetch></prefetch>` tag. Within the `<prefetch>` tag, different tags are used for each extraction. The four sub-levels are header, volume, filenames, and dirnames.

The `<header>` section describes the possible operating system the prefetch came from, the prefetch file header size, the applications name, the number of times the application was run and when it was last accessed.

The `<volume>` tag presents the serial number and name of the device from which the program was run. The accuracy of the serial number was confirmed through experiments.

The `<filenames>` and `<dirnames>` sections are the list of DLLs and directories used by the application during the startup process.

4.1 Limitations

scan_winprefetch must address the possibility of corrupted or missing data. The scanner is will continuously process the entire area that contains a prefetch file signature. Several bytes or the entire section may have been overwritten, thus removing the possibility of extracting useful information. Once the scanner is completed, *winprefetch.txt* will contain the extracted data displayed, but this may include some corrupted data. (*scan_winprefetch* contains logic to detect and suppress some kinds of corruption.) At the point that corruption is detected, the scanner stops processing the current prefetch file and starts scanning for the next.

5 Conclusion and Future Work

Prefetch files will continue to be used by Windows operating system and forensic examiners. The scanner, *scan_winprefetch*, provides *bulk_extractor* the additional ability to find prefetch files and extract artifacts relatively quickly. The artifacts from prefetch files can provide the examiner with answers to when, what and where.

```
<prefetch>
  <header>
    <os>Windows Vista or Windows 7</os>
    <header_size>240</header_size>
    <filename>IEXPLORE.EXE</filename>
    <runs>53</runs>
    <atime>2011-02-07T13:03:24</atime>
  </header>
  <volume>
    <path>\DEVICE\HARDDISKVOLUME1</path>
    <serial_number>b46f6927</serial_number>
  </volume>
  <creation>2010-08-18T06:13:10</creation>
  <filenames>
    <file>\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\NTDLL.DLL</file>
    <file>\DEVICE\HARDDISKVOLUME1\WINDOWS\SYSTEM32\KERNEL32.DLL</file>
  </filenames>
  <dirnames>
    <dir>\DEVICE\HARDDISKVOLUME1\PROGRAM FILES</dir>
    <dir>\DEVICE\HARDDISKVOLUME1\PROGRAM FILES\COMMON FILES</dir>
    <dir>\DEVICE\HARDDISKVOLUME1\PROGRAM FILES\JAVA</dir>
    <dir>\DEVICE\HARDDISKVOLUME1\PROGRAM FILES\JAVA\JRE6</dir>
  </dirnames>
</prefetch>
```

Figure 3: Pretty-printed XML output of winprefetch.txt

The *winprefetch.txt* file created by the scanner can be analyzed with automated tools or can be reviewed manually by the examiner using simple commands like *Notepad.exe* and *grep*. The content of the prefetch files can help differentiate between malicious applications, hidden applications, and trusted applications.

The prefetch scanner described here can be extended to process Superfetch files. Although they are in the same directory as the prefetch files, they have a different file format. Currently there is little use, little to no known information about superfetch and its forensic value. The behavior was briefly described by Microsft to create a profile of the users' application's activity which includes the days and times the user runs them [10]. Prefetch provides the forensic examiner of when an application was last used and how many times. The superfetch will provide a more fine grained activity timeline of an application and when the application is most often used. Windows Vista and Windows 7 creates *both* prefetch and Superfetch files.

References

[1] Mark A. Wade, *Decoding Prefetch Files For Forensic Purposes*. Harris Corporation, 2010.

[2] Luis E. Garcia II, *Prefetch_XML*. July, 2011 `http://www.forensicswiki.org/wiki/Prefetch_XML`

[3] Luis E. Garcia II, *Prefetch*. July, 2011 http://www.forensicswiki.org/wiki/Prefetch

[4] Jesse Kornblum, *My You Look SuperFetching*. DoD Cyber Crime Conference, 2008 http://jessekornblum.com/presentations/dodcc08-2.html

[5] Microsoft, Archived Paper: *Kernel Enhancements for Windows XP*. January, 2003 http://msdn.microsoft.com/en-us/windows/hardware/gg463468.aspx#ECLAC

[6] Vance Morrison, *The windows prefetcher*. April, 2007 http://blogs.msdn.com/b/vancem/archive/2007/04/15/the-windows-prefetcher.aspx

[7] JSKYS, *Prefetch Files*. June, 2011 http://members.rushmore.com/~jsky/id14.html

[8] Simson L. Garfinkel, Digital media triage with bulk data analysis and *bulk_extractor* Naval Postgradaute School. July, 2011

[9] Jesse Kornblum *Superfetch* May, 2007 http://www.forensicswiki.org/wiki/SuperFetch

[10] Ed Bott, Carl Siechert, and Craig Stinson. *Get maximum performance from Windows Vista*. Microsoft Corporation 2007 http://windows.microsoft.com/en-US/windows-vista/Get-maximum-performance-from-Windows-Vista-from-Windows-Vista-Inside-Out

[11] Yogesh Kharti *Yogesh Kharti Prefetch Research* August, 2011 http://42llc.net/?page_id=215

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Research and Sponsored Programs Office
   Naval Postgraduate School
   Monterey, California